

Khushveen Kaur Umra

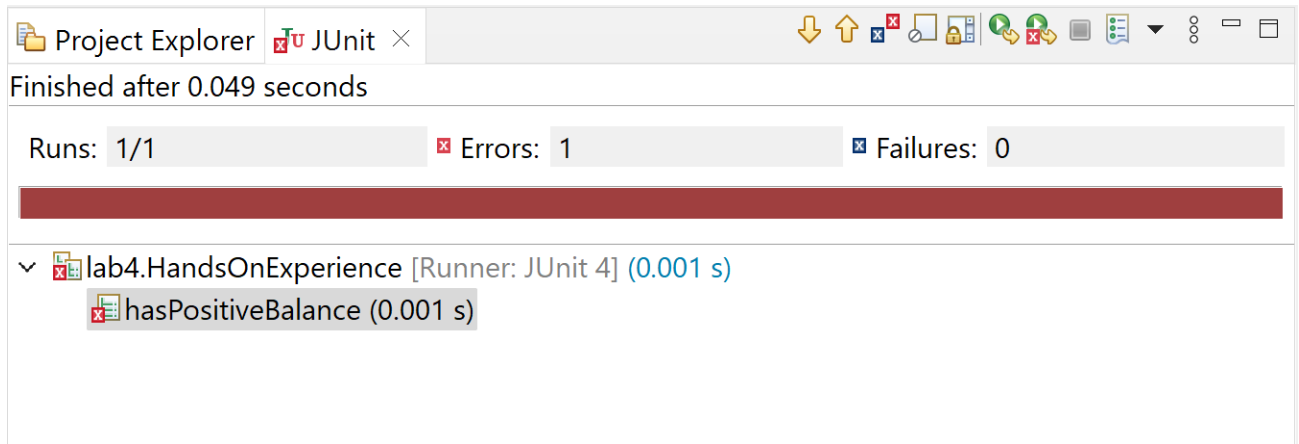
SE 317

14th June 2022

Lab 4 Submission

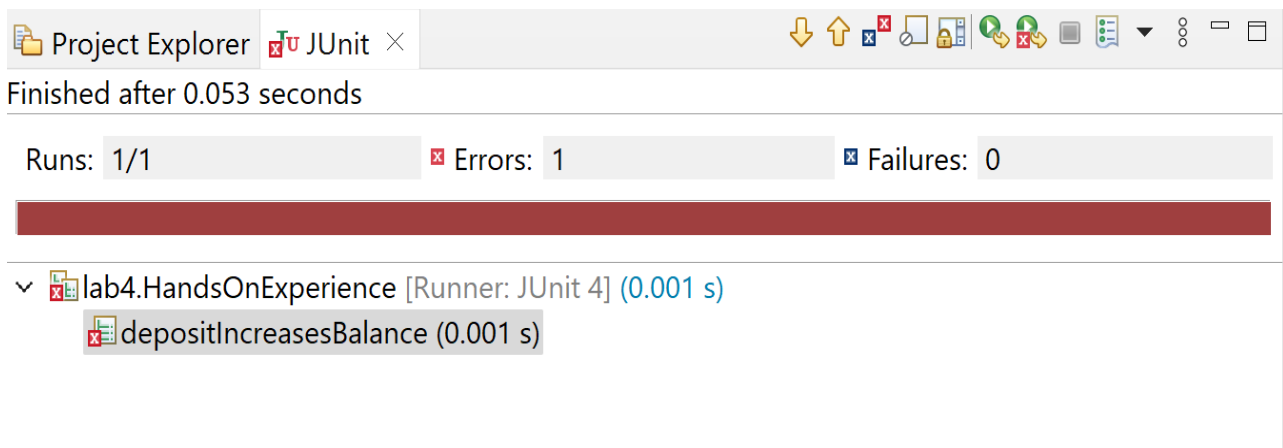
- 1) The first two test cases failed. This is because no account was created or initialized prior to these test cases, hence the deposit and the balance view test cases failed as there was no account in the system that would allow the deposit to be initialized or allow the user to know if the balance in the account is now greater than zero.

a) hasPositiveBalance test case failed:



The screenshot shows the JUnit test runner interface. At the top, it says "Project Explorer" and "JUnit". Below that, it says "Finished after 0.049 seconds". A summary bar shows "Runs: 1/1", "Errors: 1", and "Failures: 0". A red progress bar is visible. The test results list shows a failed test case: "lab4.HandsOnExperience [Runner: JUnit 4] (0.001 s)" with a sub-item "hasPositiveBalance (0.001 s)" marked with a red 'x' icon.

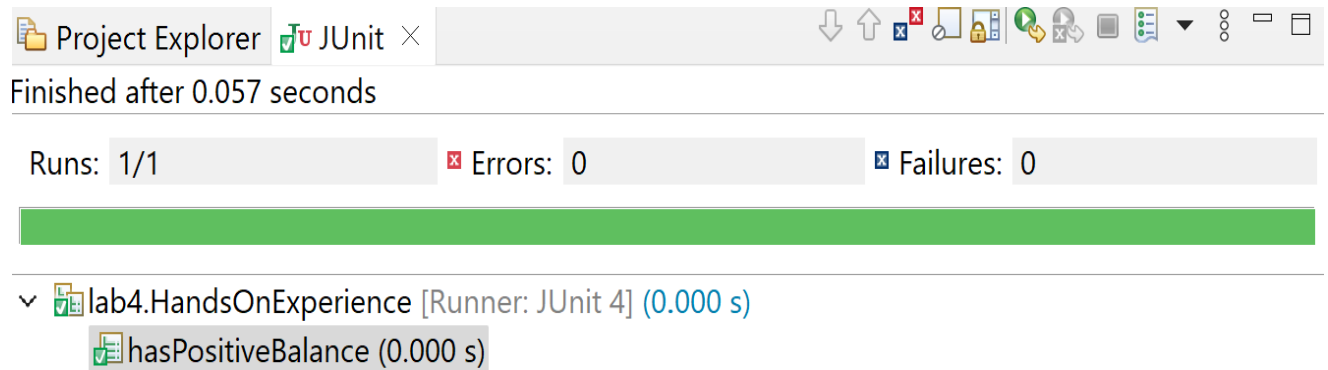
b) depositIncreasesBalance test case failed:



The screenshot shows the JUnit test runner interface. At the top, it says "Project Explorer" and "JUnit". Below that, it says "Finished after 0.053 seconds". A summary bar shows "Runs: 1/1", "Errors: 1", and "Failures: 0". A red progress bar is visible. The test results list shows a failed test case: "lab4.HandsOnExperience [Runner: JUnit 4] (0.001 s)" with a sub-item "depositIncreasesBalance (0.001 s)" marked with a red 'x' icon.

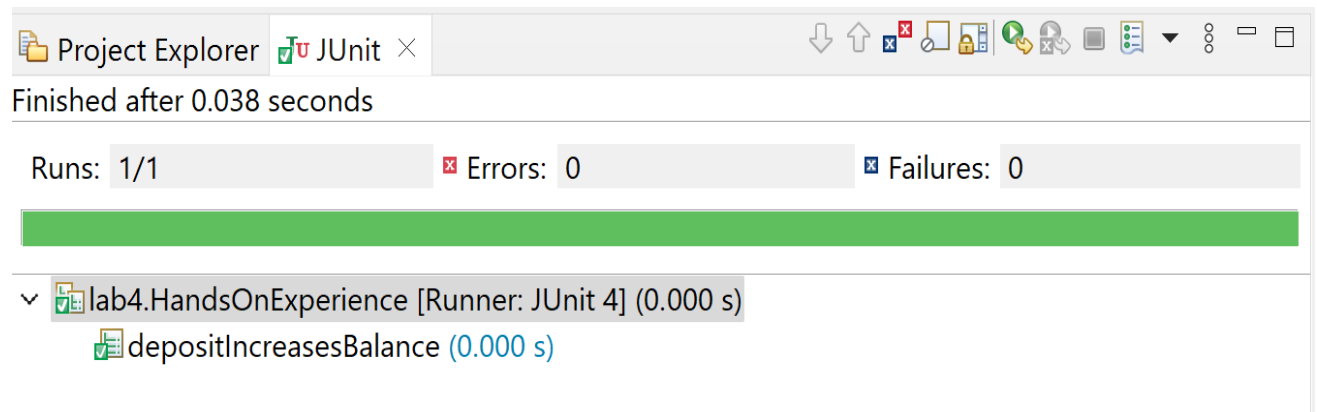
- 2) After creating or initializing an account instance, the following two test cases passed, as now there was an account where money could be deposited, and the balance could be viewed. Hence, by initializing an account, we resolved the test failures seen above.

a) hasPositiveBalance test case passed after initializing an account :



The screenshot shows the IntelliJ IDEA JUnit test runner interface. At the top, the 'Project Explorer' and 'JUnit' tabs are visible. Below the tabs, it says 'Finished after 0.057 seconds'. A summary bar shows 'Runs: 1/1', 'Errors: 0', and 'Failures: 0'. A green progress bar is at the bottom. The test results list shows a single test case: 'lab4.HandsOnExperience [Runner: JUnit 4] (0.000 s)' with a green checkmark icon, and 'hasPositiveBalance (0.000 s)' with a green checkmark icon.

b) depositIncreasesBalance test case passed after initializing an account :



The screenshot shows the IntelliJ IDEA JUnit test runner interface. At the top, the 'Project Explorer' and 'JUnit' tabs are visible. Below the tabs, it says 'Finished after 0.038 seconds'. A summary bar shows 'Runs: 1/1', 'Errors: 0', and 'Failures: 0'. A green progress bar is at the bottom. The test results list shows a single test case: 'lab4.HandsOnExperience [Runner: JUnit 4] (0.000 s)' with a green checkmark icon, and 'depositIncreasesBalance (0.000 s)' with a green checkmark icon.

- 3) The test case runs successfully, with no errors and failures. Upon inspection, it is found that it will show the comment when the test fails, as the “assertThat” function is used to check if the specified balance value matches with the expected balance value.

Finished after 0.06 seconds

Runs: 1/1 ✖ Errors: 0 ✖ Failures: 0

✓ lab4.HandsOnExperience [Runner: JUnit 4] (0.000 s)
 ✓ testWithWorthlessAssertionComment (0.000 s)

- ➔ For example, if we change the value in “equalTo()” from 50 to any other number, it will fail the test, and the comment will be viewed. Here, I tested with the number “200”.

Finished after 0.078 seconds

Runs: 1/1 ✖ Errors: 0 ✖ Failures: 1

✖ lab4.HandsOnExperience [Runner: JUnit 4] (0.001 s)
 ✖ testWithWorthlessAssertionComment (0.001 s)

≡ Failure Trace

java.lang.AssertionError: account balance is 100

Expected: <200>

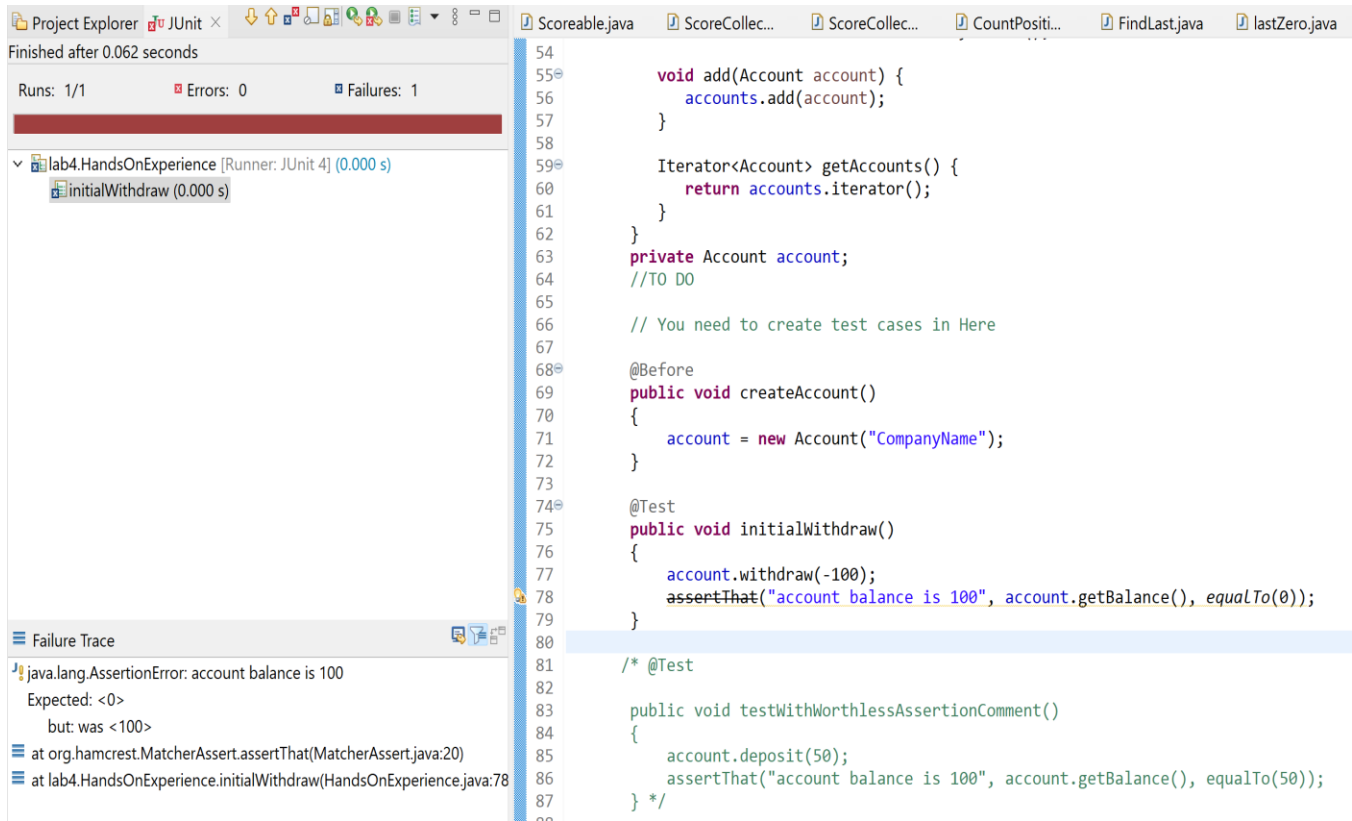
but: was <50>

at org.hamcrest.MatcherAssert.assertThat(MatcherAssert.java:20)

at lab4.HandsOnExperience.testWithWorthlessAssertionComment(HandsOnExperience.java:79)

- 4) Based on this test case, the original balance in the account should have remained as 0, as one cannot withdraw an amount equal to a negative amount. Here, the error in the code is that the function “initialWithdraw” does not check if the parameter entered for the withdraw is negative or positive. Technically the test fails because the expected value is not the same as the amount depicted. But logically, the test case should have failed because the parameter passed to withdraw was negative. Hence, since the function did not check if the parameter is positive or not, the test did not run successfully.

Moreover, the expected result should have been “0”, but the answer we got is “100”, which means that the test failed and did not run successfully. This showcases the error in the code.

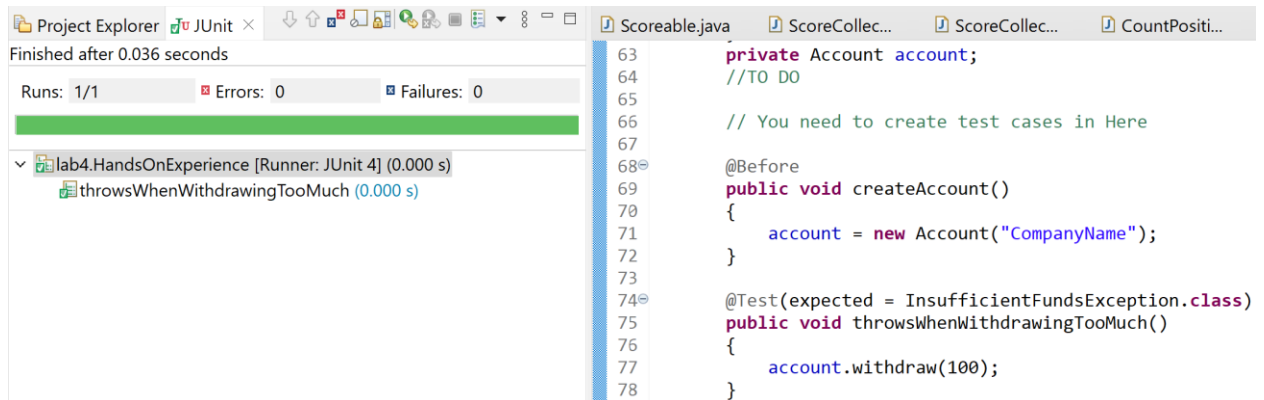


```
54
55 void add(Account account) {
56     accounts.add(account);
57 }
58
59 Iterator<Account> getAccounts() {
60     return accounts.iterator();
61 }
62
63 private Account account;
64 //TO DO
65
66 // You need to create test cases in Here
67
68 @Before
69 public void createAccount()
70 {
71     account = new Account("CompanyName");
72 }
73
74 @Test
75 public void initialWithdraw()
76 {
77     account.withdraw(-100);
78     assertThat("account balance is 100", account.getBalance(), equalTo(0));
79 }
80
81 /* @Test
82
83 public void testWithWorthlessAssertionComment()
84 {
85     account.deposit(50);
86     assertThat("account balance is 100", account.getBalance(), equalTo(50));
87 } */
```

Failure Trace

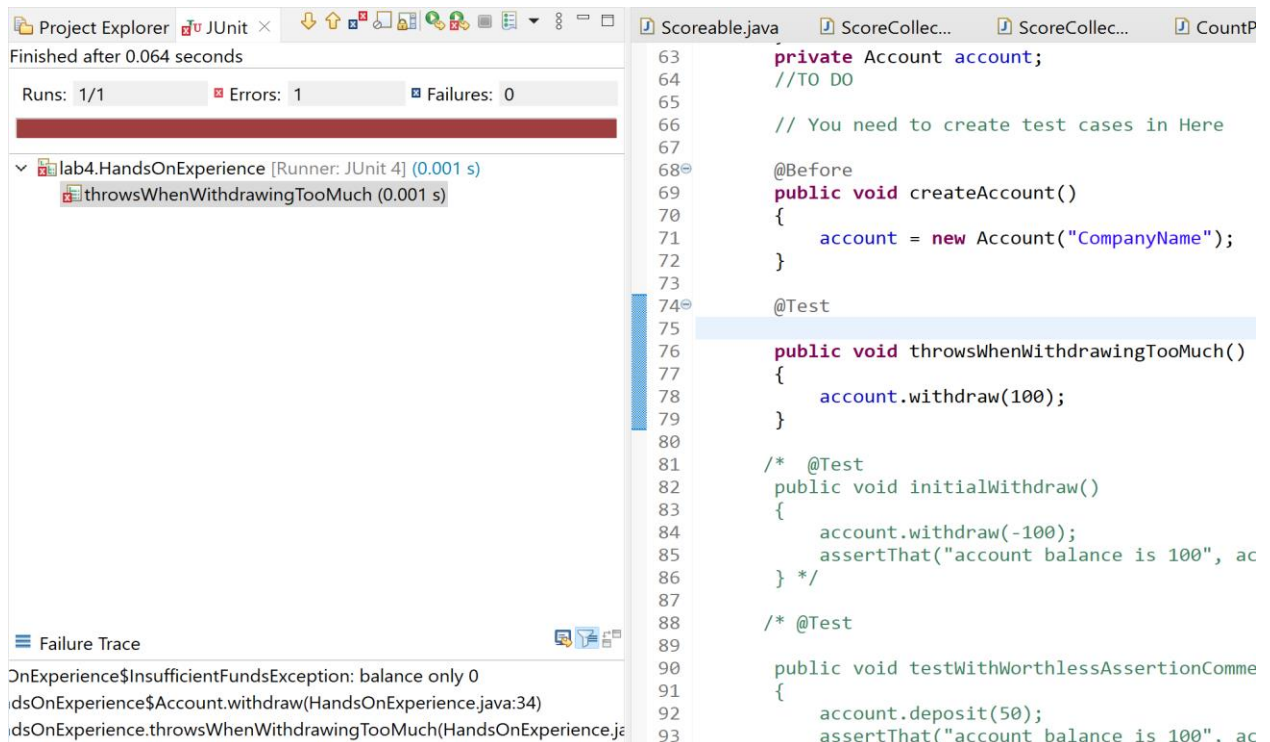
```
java.lang.AssertionError: account balance is 100
Expected: <0>
but: was <100>
at org.hamcrest.MatcherAssert.assertThat(MatcherAssert.java:20)
at lab4.HandsOnExperience.initialWithdraw(HandsOnExperience.java:78)
```

- 5) A) With the exception of insufficient funds, the test runs successfully, as it already expects that the exception is thrown and checks for it while the test runs. The exception indicates that there were insufficient funds to perform the withdrawal, and hence the test passed.



```
63     private Account account;  
64     //TO DO  
65  
66     // You need to create test cases in Here  
67  
68     @Before  
69     public void createAccount()  
70     {  
71         account = new Account("CompanyName");  
72     }  
73  
74     @Test(expected = InsufficientFundsException.class)  
75     public void throwsWhenWithdrawingTooMuch()  
76     {  
77         account.withdraw(100);  
78     }
```

B) Without the exception or the expected value, the test does not run successfully, as it indicates in the comment that the balance is 0. Without the exception, it does not know what to expect because it does not know that the withdrawal amount is greater than the available balance. That is, the program terminated the test as the line that caused the exception is not executed. Hence, the test fails in this case.



```
63     private Account account;  
64     //TO DO  
65  
66     // You need to create test cases in Here  
67  
68     @Before  
69     public void createAccount()  
70     {  
71         account = new Account("CompanyName");  
72     }  
73  
74     @Test  
75     public void throwsWhenWithdrawingTooMuch()  
76     {  
77         account.withdraw(100);  
78     }  
79  
80  
81     /* @Test  
82     public void initialWithdraw()  
83     {  
84         account.withdraw(-100);  
85         assertThat("account balance is 100", ac  
86     } */  
87  
88     /* @Test  
89  
90     public void testWithWorthlessAssertionComme  
91     {  
92         account.deposit(50);  
93         assertThat("account balance is 100", ac
```

Failure Trace

OnExperience\$InsufficientFundsException: balance only 0
dsOnExperience\$Account.withdraw(HandsOnExperience.java:34)
dsOnExperience.throwsWhenWithdrawingTooMuch(HandsOnExperience.java:78)